A DATA MANIPULATION MODEL : AN EXTENSION OF THE ALPHA EXPRESSION

AUTHOR(S):

KOBAYASHI, Isamu

CITATION:

KOBAYASHI, Isamu. A DATA MANIPULATION MODEL : AN EXTENSION OF THE ALPHA EXPRESSION. 数理解析研究所講究録 1984, 525: 159-169

ISSUE DATE:

1984-06

URL:

http://hdl.handle.net/2433/98514

数理解析研究所講究録
525巻 1984年 159-169

A DATA MANIPULATION MODEL : AN EXTENSION OF THE ALPHA EXPRESSION

Isamu KOBAYASHI

小林功武 (産能大)

SANNO Institute of Business Administration
School of Management and Informatics
Kamikasuya 1573, Isehara, Kanagawa 259-11,
JAPAN

February 1984

ABSTRACT: Two major abstract data manipulation models, the relational calculus and relational algebra, were proposed in relation to Relational Model. However, these are known to be not powerful enough for dealing with advanced applications that require various complicated operations in various value sets. Also it is known that these are not suitable for describing traditional data processing applications. In this paper two extensions of the alpha expression, a pair of a relational calculus and a target list, are proposed. One is a extension that enables us to use various operators (including aggregate operators) in various value set in defining the relational calculus and target list. The second extension is introduction of imaginary tuples that enables us easy description and effective implementation of traditional data processing applications.

KEYWORDS AND PHRASES: alpha expression, alpha operation, aggregate function, database, data manipulation, function of tuples, imaginary tuples, relational calculus, target list, tuple-generating function, tuple-selecting function.

1. INTRODUCTION

Data manipulation model has almost the same importance to the data structure model. Nevertheless the former has not been so well studied as the latter.

/

As for the relational model, Codd proposed two major abstract data manipulation models; the relational calculus which is proposed in relation to the Alpha sublanguage [Codd 1971], and the relational algebra which was proven to be relationally complete with respect to the relational calculus [Codd 1972]. However, no major enhancements have been made on these two models though these are not powerful enough for some advanced applications.

This paper presents an abstract data manipulation model which is an extension of the alpha expression that is a pair of a target list and a relational calculus. Codd's alpha expression is a special case of the abstract data manipulation model presented here. This model is much more convenient for describing various data manipulations in a wider application area.

2. DATA PROCESSING AS A SET OPERATION

Data processing, in general, can be regarded as an operation that transforms m given input into m' outputs. The unit of inputs and outputs is a tuple and several tuples are collectively treated as an input relation or output relation. The input relation can be placed on punched cards, paper tapes, magnetic tapes, magnetic disks or remote terminals connected to the host computer via a communication network. The output relation can also be placed on punched cards, paper tapes, printer sheets, magnetic tapes, magnetic disks or remote terminals. Let $R_1,R_2,\ldots,R_m$ be m input relations, and $R'_1,R'_2,\ldots,R'_{m'}$ be m' output relations. Each relation may be composed of only one tuple or more than one tuple. Then the data processing can be regarded as a transformation

$$\tau(R_1,R_2,\ldots,R_m)=(R'_1,R'_2,\ldots,R'_{m'}).$$

If we disregard the processing performance to achieve this transformation, it can obviously be decomposed into m' transformations of the form

$$\tau_k(R_1,R_2,\ldots,R_m)=R'_k \quad (k=1,2,\ldots,m')$$

This implies that the essentials of data processing is m-ary set operations. Let us next consider to formally describe such a set operation. Generally a transformation

$$\tau(R_1,R_2,\ldots,R_m)=R$$

can be defined by

$$R=\{f(x_1,x_2,\ldots,x_m)\,|\,g(x_1,x_2,\ldots,x_m)\},$$

where g is a logical function of the form

$$g(x_1,x_2,\ldots,x_m) \equiv x_1/R_1 \wedge x_2/R_2 \wedge \ldots \wedge x_m/R_m \wedge g'(x_1,x_2,\ldots,x_m)$$

with x_k/R_k being a range term that becomes true if the tuple variable x_k is substituted by a tuple t_k in R_k, and false otherwise. The function g' is an arbitrary logical function defined on $R_1 \times R_2 \times \ldots \times R_m$. It may include some other range terms x_k/R'_k or their denial $\sim x_k/R'_k$ conjunctively conbined with other terms. (It may also include other tuple variables than $x_1,x_2,\ldots,x_m$ which are bound in g' by some means.) The function g is used for selecting ordered sets of tuples qualified for g' from $R_1 \times R_2 \times \ldots \times R_m$ and called a tuple-selecting function (TSF).

On the other hand the function f is a ordered set of n functions of tuple variables $x_1,x_2,\ldots,x_m$ where n is the number of attributes in the output relation. It is used for generating a tuple in the output relation from each ordered set of tuples selected by the TSF g. The function f is called a tuple-generating function (TGF).

For example, let R_{emp} and R_{ot} be two input relations. The relation R_{emp} has three attributes emp-no, salary and ot-rate, while R_{ot} has two attributes emp-no and overtime. A relation R_{pay} with four attributes emp-no, salary, ot-charge and net-pay can be defined by the TSF

$$g(x_1,x_2) \equiv x_1/R_{emp} \wedge x_2/R_{ot} \wedge \text{emp-no}(x_1)=\text{emp-no}(x_2)$$

and the TGF

$$f(x_1,x_2) \equiv (\text{emp-no}(x_1),\text{salary}(x_1),\text{ot-charge}(x_1,x_2),\text{net-pay}(x_1,x_2))$$

where

$$\text{ot-charge}(x_1,x_2) \equiv \text{ot-rate}(x_1) \times \text{overtime}(x_2),$$

$$\text{net-pay}(x_1,x_2) \equiv \text{salary}(x_1)+\text{ot-charge}(x_1,x_2),$$

if it is assumed that every tuple in R_{emp} has a corresponding tuple (a tuple with the same emp-no value) in R_{ot}.

Let us call an operation

$$a[f:g](R_1,R_2,\ldots,R_m)=\{f(x_1,x_2,\ldots,x_m) \mid g(x_1,x_2,\ldots,x_m)\}$$

an alpha operation with alpha expression $f:g$ after Codd's Alpha sublanguage. An alpha expression in Alpha sublanguage is a pair composed of a relational calculus and a target list. We are considering a wider class of alpha expressions. The relational calculus and target list are respectively special TSFs and TGFs. Let us next consider how these TSFs and TGFs must be defined in a formal way.

3. FUNCTION OF TUPLES

There are various ways of defining TSFs and TGFs; however, we will start

at defining function of tuples which become the basis of describing TSFs and TGFs. Function of tuples (FOTs) are functions

$$\phi: R_1 \times R_2 \times \ldots \times R_m \rightarrow V_\phi$$

where $R_1, R_2, \ldots, R_m$ are m relations and V_ϕ is an arbitrary value set. The above FOT ϕ is said to be of span m. As a special case the span m can be 0.

FOTs are defined by FOT1 through FOT6 described below.

(FOT1) For an arbitrary constant C, $\phi(\ ) \equiv C$ is an FOT of span 0 (with no tuple variables).

(FOT2) For an attribute A_k, $\phi(x) \equiv A_k(x)$ is a function of tuples of span 1 (with a single tuple variable x). If the attribute A_k is undefined for the relation R over which x is ranged, then the $A_k(x)$ value is assumed to be Ω (undefined).

(FOT3) A range term $\phi(x) \equiv x/R$ is an FOT of span 1. Its value becomes true when x is substituted by a tuple belonging to R, and false otherwise.

Given a relation instance R, the values of functions defined by FOT2 and FOT3 can be directly evaluated. Conversely, only the FOTs defined by FOT1 through FOT3 can be directly evaluated provided that a relation instance R is given. In this sense the FOTs defined by FOT1 through FOT3 are collectively called basic FOTs (BFOTs).

Let z be an ordered set of tuple variables $(x_1, x_2, \ldots, x_m)$. Let us assume that it is possible to write $\phi(x_1, x_2, \ldots, x_m)$ simply as $\phi(z)$. Also let us denote $\{x_1, x_2, \ldots, x_m\}$ by $\bar{z}$.

(FOT4) Let $\phi_k(z_k)$ (k=1,2,...,m) be an FOT whose range is a value set V_k, and V_ψ is an arbitrary value set. If an operator

$$\psi: V_1 \times V_2 \times \ldots \times V_m \rightarrow V_\psi$$

is defined over these value sets, then the function $\bar{\psi}(\phi_1, \phi_2, \ldots, \phi_m)$ defined by

$$\bar{\psi}(\phi_1, \phi_2, \ldots, \phi_m)(z) \equiv \psi(\phi_1(z_1), \phi_2(z_2), \ldots, \phi_m(z_m))$$

is an FOT. Here $\bar{z} = \bigcup_{k=1}^{m} \bar{z}_k$. The span of this FOT is the cardinality of $\bar{z}$.

Provided that BFOTs are typed, that is, BFOTs are classfied into several groups according to their ranges, operators defined over various value sets can be extended to the operators that conbine FOTs to form new FOTs. Let Φ_k be the set of FOTs with the range V_k. Then an operator

$$\psi: V_1 \times V_2 \times \ldots \times V_m \rightarrow V_\psi$$

is extended to the operator

$\bar{\psi}:\Phi_1\times\Phi_2\times\ldots\times\Phi_m\to\Phi_\psi$

where Φ_ψ is a set of FOTs with the range V_ψ. Usually the same notation is used for both ψ and $\bar{\psi}$. If an infix notation is used, parentheses must be introduced to specify the operator precedence.

It is not necessary to restrict attributes to those with simple value sets. The attribute can be of structured type like fixed or variable length arrays, sets (power set elements), graphs, records (Cartesian product elements) and repeating groups (elements of a power set of a Cartesian product). Variuos operators are defined in these value sets. The operators ψ can not only be common operators like arithmetic, logical and relational operators defined in simple sets, but also be various complicated operators like addition and inner product (vectors), addition, multiplication, determinant and eigenvalue (matrices), pop, push and top (stacks), enqueue and dequeue (queues), union, intersection, difference, belongs to, included in and intersect (power set members), concatenation and partial match (graphs) defined in compound value sets. All these operators can be extended to the operators $\bar{\psi}$ defined in the corresponding sets of FOTs. To cope with a wider class of applications that require such complicated operators, it is not advisable to impose the first normal form transformation on the relation schema.

(FOT5) Let $\phi_1(x_1,x_2,\ldots,x_m,x_{m+1},\ldots,x_p)$ be an FOT of the form

$$\phi_1(x_1,x_2,\ldots,x_m,x_{m+1},\ldots,x_p)$$
$$\equiv x_1/R_1\wedge x_2/R_2\wedge\ldots\wedge x_m/R_m\wedge\phi'(x_1,x_2,\ldots,x_m,x_{m+1},\ldots,x_p)$$

where ϕ' is an FOT whose range is the truth value set (logical FOT) which does not contain any range terms regarding tuple variables $x_{m+1},x_{m+2},\ldots,x_p$. For tuple variables $x_1,x_2,\ldots,x_m$, ϕ' may contain range terms x_k/R_k' or their denial $\sim x_k/R_k'$ conjunctively conbined to other terms. Let $\phi_2(x_1,x_2,\ldots,x_m,x_{p+1},\ldots,x_q)$ be an FOT whose range is V. If an aggregate function ∇ is defined in the value set V, that is, if

$$\nabla:\bigcup_{k=1}^{\infty}V^k\to V,$$

then $\bar{\nabla}[\phi_1]\phi_2$ defined by

$$(\bar{\nabla}[\phi_1]\phi_2)(z)$$
$$\equiv\nabla[\phi_1(x_1,x_2,\ldots,x_m,x_{m+1},\ldots,x_p)]\phi_2(x_1,x_2,\ldots,x_m,x_{p+1},\ldots,x_q)$$

is an FOT whose range is V. Here $\bar{z}=\{x_{m+1},x_{m+2},\ldots,x_p\}\cup\{x_{p+1},x_{p+2},\ldots,x_q\}$.

The right hand side of the above definition means that the $\phi_2(t_1,t_2,\ldots,t_m,x_{p+1},\ldots,x_q)$ values are aggregated for the ordered sets $(t_1,t_2,\ldots,$

t_m) of tuples qualified for $\phi_1(t_1,t_2,\ldots,t_m,x_{m+1},\ldots,x_p)$. Either or both $\{x_{m+1},x_{m+2},\ldots,x_p\}$ and $\{x_{p+1},x_{p+2},\ldots,x_q\}$ can be an empty set. If the both sets are empty, $\bar{z}=\emptyset$ and $\bar{\nabla}[\phi_1]\phi_2$ becomes constant function.

Let Φ_L be the set of FOTs that satisfy conditions imposed on the ϕ_1 defined above, and Φ_V with the set of FOTs whose range is V. Then $\bar{\nabla}$ is an operator

$$\bar{\nabla}:\Phi_L\times\Phi_V\rightarrow\Phi_V.$$

Usually the same notation is used for representing ∇ and $\bar{\nabla}$.

Tuple variables $x_1,x_2,\ldots,x_m$ are said to be bound in the scope of the aggregate operator ∇, while tuple variables in $\bar{z}$ are said to be free in the scope of ∇.

There are various aggregate operators defined on various value sets. The aggregate operator Σ, Π, max, min, average and standard deviation are defined on the set of numbers. The aggregate operator Σ corresponds to the arithmetic operator +, while Π corresponds to $\times$. In a similar way, it is possible to introduce an aggregate operator $\bigwedge$ corresponding to $\wedge$ and $\bigvee$ corresponding to $\vee$ in the truth value set. These two operators are different from others in the point that the range of ϕ_2 is the truth value set as well as that of ϕ_1. It is easy to see that

$$\bigwedge[x/R\wedge\phi_1]\phi_2\equiv\bigwedge[x/R](\phi_1\supset\phi_2)$$

and

$$\bigvee[x/R\wedge\phi_1]\phi_2\equiv\bigvee[x/R](\phi_1\wedge\phi_2).$$

In particular, $\bigwedge[x/R]\phi$ is written as $(\forall x/R)\phi$, while $\bigvee[x/R]\phi$ as $(^{\exists}x/R)\phi$. The tuple variable x is said to be quantified by the universal quantifier $\forall$ or by the existential quantifier $\exists$. Obviously quantified variables are bound variables. This is an interpretation of quantifications as aggrerate functions. In database environment where relations are composed of a finite number of tuples, such an interpretation often becomes useful [Kobayashi 1984].

(FOT6) Only functions defined by FOT1 through FOT5 are FOTs.

Now we are ready to define TSFs and TGFs.

(TSF) A TSF is an FOT of the form

$$g(x_1,x_2,\ldots,x_m)\equiv x_1/R_1\wedge x_2/R_2\wedge\ldots\wedge x_m/R_m\wedge\phi(x_1,x_2,\ldots,x_m),$$

where ϕ is an FOT whose range is the truth value set. Tuple variables $x_1,x_2,\ldots,x_m$ are free in ϕ, that is, they are not bound in the scope of any aggregate operators used in defining ϕ. The FOT ϕ may contain range terms x_k/R_k' or their denial $\sim x_k/R_k'$ for free variables x_k if these are conbined conjunctively to other terms. It may also contain

other tuple variables than $x_1, x_2, \ldots, x_m$, which are bound in the scope of some aggregate operators used in defining ϕ.

As TSFs of a special type, we may consider TSFs with no free variables. Such a TSF becomes constant true or false according to the state of relations over which tuple variables are bound. Such TSFs are regarded as rules imposed on database relations. These rules may be used as integrity constraints against which database updates must be validated, or as axioms to be used in deductive question answering.

(TGF) A TGF is an FOT of the form

$$f(z) \equiv (\phi_1(z), \phi_2(z), \ldots, \phi_n(z)),$$

where $\phi_1, \phi_2, \ldots, \phi_n$ are FOTs.

A TGF is an ordered set of n FOTs. Given an ordered set z of tuples, it is used to generate a tuple with n attributes.

Let $S(F_s, \Phi_s, G_s)$ be a set of TSFs generated from a set F_s of BFOTs using a set Φ_s of operators according to a generative grammar G_s, and $G(F_g, \Phi_g, G_g)$ be the set of TGFs generated from a set F_g of BFOTs using a set Φ_g of operators according to a generative grammar G_g. The sets S and G can be regarded as representing the total data processing capability possessed by a given system. In this section a very wide class of TSFs and that of TGFs have been presented; however, in most cases (data models and database management systems) only TSFs and TGFs of a restricted form are dealt with.

In information algebra [CODASYL 1962], an alpha operation is called a bundling operation. A TSF and a TGF were respectively called a bundling function and a function of bundles. As F_s and F_g, FOTs defined by FOT1 through FOT3 were used. However, only a very limited number of operators were used, and the grammars G_s and G_g were defined in a very conventional manner.

As mentioned previously in Relational Model, a TSF and a TGF were respectively called a relational calculus and a target list. As F_s, FOTs defined by FOT1 through FOT3 were used. However, as operators in Φ_s, only six relational operators and universal and existential quantifiers were used. The grammar G_s was almost the same to that mentioned in this section. On the other hand, only FOTs defined by FOT2 were used as F_g, and Φ_g was an empty set. The grammar G_g was a very restricted one accordingly.

To accord with a wider class of applications including advanced engineering applications, it is very desirable to deal with the alpha operations with extended alpha expressions described here.

4. IMAGINARY TUPLES

The alpha operation so far defined is very powerful to describe data processing applications in general. However, it still has a fatal disadvantage when its being used to describe traditional business data processing application.

Let us reconsider the example presented in section 2. There were two input relations R_{emp} and R_{ot}, for which it is assumed that every tuple in R_{emp} has its corresponding tuple in R_{ot}, that is, for every tuple t_1 in R_{emp} there exist one (and only one) tuple t_2 in R_{ot} for which emp-no(t_1) =emp-no(t_2). However in practical applications, there can be a tuple in R_{emp} for which no corresponding tuples exist in R_{ot}, and also there can be a tuple in R_{ot} for which no corresponding tuples exist in R_{emp}. (The latter may be an erroneous case.)

Selecting a tuple in R_{emp} with a corresponding tuple in R_{ot} can be made by specifying the TSF

$$g_1(x_1,x_2) \equiv x_1/R_{emp} \wedge x_2/R_{ot} \wedge \text{emp-no}(x_1)=\text{emp-no}(x_2).$$

The TGF for this case is

$$f_1(x_1,x_2) \equiv (\text{emp-no}(x_1),\text{salary}(x_1),\text{ot-charge}(x_1,x_2),\text{net-pay}(x_1,x_2),)$$

where

$$\text{ot-charge}(x_1,x_2) \equiv \text{ot-rate}(x_1) \times \text{overtime}(x_2)$$

and

$$\text{net-pay}(x_1,x_2) \equiv \text{salary}(x_1)+\text{ot-charge}(x_1,x_2).$$

Selecting a tuple in R_{emp} with no corresponding tuple in R_{ot} can be made by specifying the TGF

$$g_2(x_1) \equiv x_1/R_{emp} \wedge (\forall x_2/R_{ot})\text{emp-no}(x_1) \neq \text{emp-no}(x_2).$$

The TGF for this case is

$$f_2(x_1) \equiv (\text{emp-no}(x_1),\text{salary}(x_1),0,\text{salary}(x_1),).$$

Finally selecting a tuple in R_{ot} with no corresponding tuple in R_{emp} can be made by specifying the TGF

$$g_3(x_2) \equiv x_2/R_{ot} \wedge (\forall x_1/R_{emp})\text{emp-no}(x_1) \neq \text{emp-no}(x_2).$$

The TGF for this case is

$$f_3(x_2) \equiv (\text{emp-no}(x_2),,,,\text{error}).$$

To create R_{pay} properly, it is necessary to execute a program like

begin R_1:=a[f_1:g_1](R_{emp},R_{ot});
R_2:=a[f_2:g_2](R_1);
R_3:=a[f_3:g_3](R_{ot});
R_{pay}:=u(R_1,R_2,R_3) end;

where u is the operation making a union of three relations R_1,R_2 and R_3. The union operation itself must be described in a procedural conbination of several alpha operations [Kobayashi 1983]. In general, given n input relations, it is necessary to execute 2^n-1 alpha operations and a union operation that conbines 2^n-1 intermediate results.

Such a program has two major problems. First it is not easy to describe a proper TSF and a TGF for each of 2^n-1 (complete or partial) matches. In fact, it is necessary to introduce one or more quantified variables. Secondly the above program, if executed as it were, includes many duplicate operations. It is known that, if R_{emp} and R_{ot} are organized in the sequence of emp-no values, the desired processing can be achieved by a single sequential collation with no redundant operations.

To resolve this difficulty it is desirable to devise some means by which the whole operation can be described (nonprocedurally) by a sinble alpha operation, which also enables an easy application of sequential collation. This is achieved by introducing imaginary tuples defined below.

An imaginary tuple i to be attached to relation R_k is defined by the following three conditions:

(IT1) $g(t_1,t_2,\ldots,t_{k-1},i,t_{k+1},\ldots,t_m)$ becomes true if and only if
$(x_k/R_x)g(t_1,t_2,\ldots,t_{k-1},x_k,t_{k+1},\ldots,t_m)$
is false but if for appropriate tuple t_k which is not currently contained in R_k,
$g(t_1,t_2,\ldots,t_{k-1},t_k,t_{k+1},\ldots,t_m)$
becomes true.

(IT2) $g(i,i,\ldots,i)$ is always false. That is, at least one tuple must be an existing tuple.

(IT3) There is an FOT f' of span m-1 such that
$f'(x_1,x_2,\ldots,x_{k-1},x_{k+1},\ldots,x_m)\equiv f(x_1,x_2,\ldots,x_{k-1},i,x_{k+1},\ldots,x_m)$.

Let us denote $R_k \cup \{i\}$ by $\hat{R}_k$. Then the above example can be achieved by the program

begin R_{pay}:=a[f:g]($\hat{R}_{emp}$,$\hat{R}_{ot}$) end;

where

$$g(x_1,x_2)\equiv x_1/\hat{R}_{emp} \wedge x_2/\hat{R}_{ot} \wedge \text{emp-no}(x_1)=\text{emp-no}(x_2)$$

$$f(x_1,x_2)\equiv \begin{cases} f_1(x_1,x_2) & (\text{if } x_1\neq i \wedge x_2\neq i) \\ f_2(x_1) & (\text{if } x_1\neq i \wedge x_2=i) \\ f_3(x_2) & (\text{if } x_1=i \wedge x_2\neq i) \end{cases}$$

with f_1, f_2 and f_3 being those defined previously.

It is not necessary to add imaginary tuples to all input relations.

For example, if a tuple in R_{ot} has always a corresponding tuple in R_{emp}, then the program can be

begin R_{pay}:=a[f:g](R_{emp},$\hat{R}_{ot}$) end;

where

$$g(x_1,x_2) \equiv x_1/R_{emp} \wedge x_2/\hat{R}_{ot} \wedge \text{emp-no}(x_1)=\text{emp-no}(x_2)$$

$$f(x_1,x_2) \equiv \begin{cases} f_1(x_1,x_2) & (\text{if } x_2 \neq i) \\ f_2(x_1) & (\text{if } x_2 = i) \end{cases}$$

The outer join introduced by Codd [Codd 1979] is a special alpha operation in which both of two input relations contain an imaginary tuple.

5. CONCLUSION

The alpha operation (specified by a pair of relational calculus and a target list) in the Alpha sublanguage was an outstanding data manipulation model proposed in relation to the Relational Model. However, since the alpha expression in the Alpha sublanguage was of a restricted form, it can hardly describe advanced applications which require complicated data manipulating operations. Also since the Alpha sublanguage was designed to describe rather simple applications for casual users, it is not suitable for dealing with traditional data processing applications.

In this paper, it is shown that the first difficulty can be resolved by extending the syntax of defining alpha expressions. In particular, FOT4 and FOT5 that enables us to use various operations already defined in variuos value sets in defining FOTs is a very powerful extension for advanced applications. This policy was partially enbodied in an experimental database management system FORIMS [Kohri 1975].

The second difficulty can be resolved by introducing imaginary tuples. This function can be easily integrated in the query optimization algorisms [Kobayashi 1981]. It is possible to extend the relational algebra to make it relationally complete with respect to the extended alpha operation [Kobayashi 1983]. Also a summary operation can be formally defined by using aggregate operations defined by FOT5. Regarding quantifiers as aggregate operators is useful to develop an optimal mechanism for validating database updates [Kobayashi 1984].

REFERENCES

[CODASYL 1962] CODASYL Development Committee, An Information Algebra:

Phase I Report, *Comm. ACM*, Vol.5, No.4, pp.190-204, 1962.

[Codd 1971] E.F.Codd, A Data Base Sublanguage Founded on the Relational Calculus, *Proc. ACM SIGFIDET '71*, pp.35-68, 1971.

[Codd 1972] E.F.Codd, Relational Completeness of Data Base Sublanguage, in R.Rustin, ed., pp.65-98, Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[Codd 1979] E.F.Codd, Extending the Data Base Relational Model to Capture More Meaning, *ACM Trans. Database Systems*, Vol.4, No.4, pp.397-434, 1979.

[Kobayashi 1981] I.Kobayashi, Evaluation of Queries Based on the Extended Relational Calculi, *Int. Jour. Computer and Information Sciences*, Vol.10, No.2, pp.63-102, 1981.

[Kobayashi 1983] I.Kobayashi, Abstract Database Operations, in T.Kitagawa, ed., Japan Annual Reviews in Electronics, Computers and Telecommunications Vol.7: Computer Science and Technologies, pp.242-261, North-Holland, Amsterdam/Ohm, Tokyo, 1983.

[Kobayashi 1984] I.Kobayashi, Validating Database Updates, to appear in *Int. Jour. Information Systems*, Vol.9, No.2, 1984.

[Kohri 1975] K.Kohri, and Y.Chiba, FORIMS Phase 2 Design Specification: A FORTRAN Oriented Information Management System, *The Soken Kiyo*, Vol.5, No.1, pp.177-210, Nippon Univac Sogo Kenkyusho, Inc., 1975.